Appendix

A PyTorch implementation of SPARO

We provide an example implementation of SPARO in PyTorch [14] as follows:

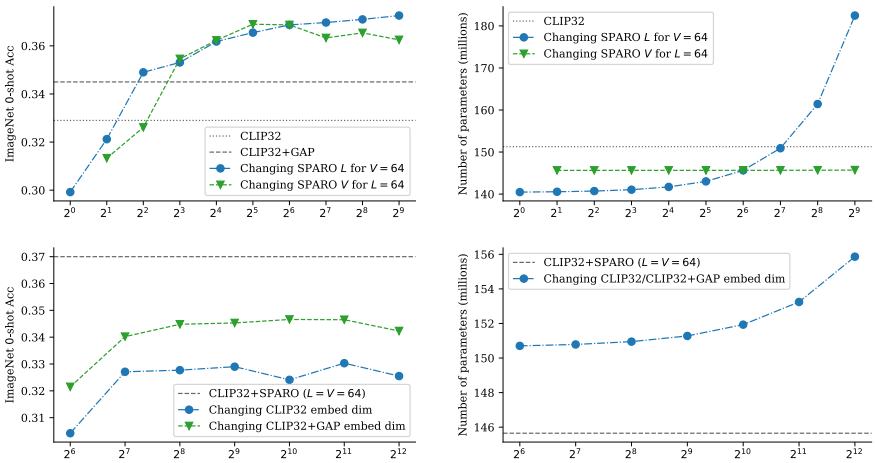
```
class Sparo(nn.Module):
    def __init__(self, d, L, V, D, grp_size=1):
        """See Sec. 3 for descriptions of d, L, V, D.
        Each key/value projection weight is shared between grp_size slots."""
        super().__init__()
        self.L, self.V, self.D = L, V, D
        self.grp_size, self.nkeys = grp_size, L // grp_size
        self.scale = D ** -0.5

        self.q = nn.Parameter(torch.randn(1, self.nkeys, grp_size, 1, D))
        self.KVproj = nn.Linear(d, self.nkeys * D)
        self.Wproj = nn.Linear(D, V)

        nn.init.xavier_uniform_(self.KVproj.weight)
        nn.init.zeros_(self.KVproj.bias)
        nn.init.zeros_(self.Wproj.bias)

    def forward(self, x, eos_idx=None):
        """Provide eos_idx for text to prevent attending to positions after it."""
        batch_size, n, d = x.shape
        kv = self.KVproj(x).view(batch_size, n, self.nkeys, 1, self.D)
        kv = kv.expand(-1, -1, -1, self.grp_size, -1).permute(0, 2, 3, 1, 4)

        attn = (self.q * self.scale) @ kv.transpose(-2, -1)
        if eos_idx is not None:
            attn_mask = (
                torch.arange(n, device=eos_idx.device)[None, None, None, None, :]
                > eos_idx[:, None, None, None, None]
            )
            attn_mask = torch.where(
                attn_mask, -torch.inf, torch.zeros_like(attn_mask, dtype=attn.dtype)
            )
            attn += attn_mask
        attn = attn.softmax(dim=-1)
        x = (attn @ kv).view(batch_size, self.L, self.D)
        return self.Wproj(x)
```

Here, we optionally utilize multi-query attention [18] to reduce the model size further by setting `grp_size`>1, allowing larger values of L with the same resources. For example, when `grp_size`=2 and `L`=64, only 32 $\boldsymbol{K}_l$ parameters are created. We encourage practitioners to explore optimal `grp_size` settings for their applications.

B Additional results

B.1 Hyperparameters L and V

SPARO encodings are concatenations of L separately-attended slots of V dimensions each. We study the effect of L and V with CLIP^{32}+SPARO trained on CC15M, comparing its ImageNet zero-shot accuracy and model size with that of



(a) ImageNet zero-shot accuracy. **(b)** Number of model parameters.

Fig. B.1: Effect of varying L and V values for CLIP^{32}+SPARO, and the embedding size for CLIP^{32} and CLIP^{32}+GAP, when training on CC15M.

CLIP^{32} and CLIP^{32}+GAP in Fig. B.1. Generally, performance is poor for very low values of L and V, and improves when they are increased. Improvements remain monotonic with L in our experimental range, but the returns diminish for larger values of L. For each increment of L, we add additional $\boldsymbol{q}_l$ and $\boldsymbol{K}_l$ parameters to the model, and considering the trade-off between model size and performance gains becomes vital. Unlike L, increasing V beyond optimal values leads to a performance drop. With very large values of V, we produce low-rank slots that can potentially encode arbitrary harmful biases for downstream tasks.

B.2 Comparison with other bottlenecks

SPARO imposes a representational bottleneck by constraining each slot to the output of a single head of attention with fixed queries and limited dimensionality. In this section, we compare our bottleneck with two other representational bottlenecks: a linear (affine) information bottleneck and a discrete bottleneck using Finite Discrete Tokens (FDT) [2].

Setup. We add the considered bottlenecks and SPARO to CLIP^{32} and CLIP^{16}, and train on different sizes of Conceptual Captions. We empirically verified that removing a block of the transformer backbone reduces the performance for all models. Still, we follow the typical setup of replacing the last transformer block of the backbone when using SPARO. However, we do not handicap other bottleneck baselines, adding the corresponding bottlenecks after the last transformer block.

Linear information bottleneck details. We implement a linear information bottleneck by setting the embedding size of standard CLIP to m and adding

Table B.1: Comparing zero-shot recognition, robustness, and SugarCrepe compositionality for CLIP models trained on Conceptual Captions with GAP, attentional pooling [9,21], a linear (affine) information bottleneck, and a discrete bottleneck using Finite Discrete Tokens (FDT) [2].

Train	Model	ImageNet-					Object Net	Sugar Crepe
		V1	V2	Sketch	A	R		
CC3M	$\text{CLIP}^{32\ (\mathcal{C})}$	0.109	0.091	0.053	0.025	0.154	0.064	0.578
	$\mathcal{C}$+GAP	0.124	0.105	0.064	0.025	0.178	0.078	0.582
	$\mathcal{C}$+AttnPool	0.114	0.092	0.045	0.026	0.149	0.070	0.589
	$\mathcal{C}$+LinearBotneck (64 → 512)	0.097	0.083	0.039	0.025	0.126	0.060	0.578
	$\mathcal{C}$+LinearBotneck (512 → 4096)	0.108	0.088	0.047	0.026	0.139	0.057	0.591
	$\mathcal{C}$+DiscreteBotneck (FDT)	0.123	0.101	0.059	**0.028**	0.166	0.069	0.583
	$\mathcal{C}$+SPARO	**0.132**	**0.113**	**0.072**	0.027	**0.190**	**0.083**	**0.596**
	$\text{CLIP}^{16\ (\mathcal{C})}$	0.141	0.122	0.068	0.033	0.177	0.080	0.611
	$\mathcal{C}$+GAP	0.156	0.134	0.069	0.033	0.187	0.081	0.616
	$\mathcal{C}$+AttnPool	0.161	0.138	0.074	0.036	0.199	0.090	0.599
	$\mathcal{C}$+LinearBotneck (64 → 512)	0.127	0.110	0.052	0.028	0.143	0.070	0.601
	$\mathcal{C}$+LinearBotneck (512 → 4096)	0.138	0.118	0.058	0.032	0.162	0.079	0.590
	$\mathcal{C}$+DiscreteBotneck (FDT)	0.161	0.137	0.073	**0.037**	0.195	**0.099**	0.600
	$\mathcal{C}$+SPARO	**0.170**	**0.140**	**0.088**	0.035	**0.221**	0.098	**0.625**
CC12M	$\text{CLIP}^{32\ (\mathcal{C})}$	0.304	0.257	0.201	0.060	0.405	0.160	0.691
	$\mathcal{C}$+GAP	0.320	0.272	0.221	0.063	0.426	0.178	0.692
	$\mathcal{C}$+AttnPool	0.290	0.240	0.170	0.046	0.357	0.165	0.679
	$\mathcal{C}$+LinearBotneck (64 → 512)	0.272	0.232	0.163	0.052	0.338	0.149	0.675
	$\mathcal{C}$+LinearBotneck (512 → 4096)	0.285	0.238	0.178	0.052	0.353	0.151	0.683
	$\mathcal{C}$+DiscreteBotneck (FDT)	0.307	0.264	0.206	0.063	0.424	0.180	**0.693**
	$\mathcal{C}$+SPARO	**0.344**	**0.293**	**0.251**	**0.069**	**0.468**	**0.200**	**0.693**
	$\text{CLIP}^{16\ (\mathcal{C})}$	0.361	0.311	0.249	0.091	0.467	0.218	0.697
	$\mathcal{C}$+GAP	0.382	0.330	0.262	0.101	0.501	0.241	0.695
	$\mathcal{C}$+AttnPool	0.357	0.305	0.226	0.085	0.428	0.226	0.685
	$\mathcal{C}$+LinearBotneck (64 → 512)	0.331	0.286	0.198	0.073	0.387	0.187	0.674
	$\mathcal{C}$+LinearBotneck (512 → 4096)	0.344	0.292	0.226	0.081	0.428	0.212	0.688
	$\mathcal{C}$+DiscreteBotneck (FDT)	0.367	0.313	0.260	0.086	0.490	0.238	0.719
	$\mathcal{C}$+SPARO	**0.406**	**0.350**	**0.298**	**0.113**	**0.559**	**0.268**	**0.723**
CC15M	$\text{CLIP}^{32\ (\mathcal{C})}$	0.329	0.279	0.232	0.070	0.435	0.178	0.687
	$\mathcal{C}$+GAP	0.345	0.294	0.242	0.077	0.460	0.192	0.678
	$\mathcal{C}$+AttnPool	0.312	0.262	0.190	0.065	0.374	0.168	0.671
	$\mathcal{C}$+LinearBotneck (64 → 512)	0.300	0.251	0.187	0.054	0.360	0.159	0.665
	$\mathcal{C}$+LinearBotneck (512 → 4096)	0.313	0.265	0.203	0.064	0.383	0.165	0.663
	$\mathcal{C}$+DiscreteBotneck (FDT)	0.333	0.282	0.233	0.071	0.456	0.197	0.685
	$\mathcal{C}$+SPARO	**0.370**	**0.313**	**0.271**	**0.083**	**0.510**	**0.213**	**0.710**
	$\text{CLIP}^{16\ (\mathcal{C})}$	0.384	0.337	0.268	0.105	0.503	0.238	0.699
	$\mathcal{C}$+GAP	0.399	0.343	0.287	0.114	0.531	0.252	0.701
	$\mathcal{C}$+AttnPool	0.387	0.329	0.248	0.100	0.469	0.233	0.685
	$\mathcal{C}$+LinearBotneck (64 → 512)	0.357	0.306	0.226	0.085	0.422	0.209	0.672
	$\mathcal{C}$+LinearBotneck (512 → 4096)	0.372	0.321	0.245	0.094	0.455	0.218	0.683
	$\mathcal{C}$+DiscreteBotneck (FDT)	0.394	0.338	0.279	0.124	0.531	0.254	0.719
	$\mathcal{C}$+SPARO	**0.437**	**0.378**	**0.317**	**0.145**	**0.579**	**0.279**	**0.730**

Table B.2: Comparing zero-shot recognition, robustness, and SugarCrepe compositionality for CLIP models trained on CC15M with ResNet-50 encoders.

Model	Params	ImageNet-					Object Net	Sugar Crepe
		V1	V2	Sketch	A	R		
CLIP^{R50} ($M = 1024$)	102M	0.298	0.272	0.227	0.090	0.430	0.220	0.679
CLIP^{R50} ($M = 4096$)	110M	0.295	0.275	0.226	0.086	0.422	0.213	0.683
CLIP^{R50}+SPARO ($L = 128, V = 32, D = 32$)	97M	0.338	0.297	**0.263**	**0.101**	0.476	**0.256**	0.713
CLIP^{R50}+SPARO ($L = 128, V = 32, D = 64$)	108M	**0.344**	**0.304**	**0.263**	0.093	**0.477**	0.252	**0.719**

an affine transformation to an output size of M on top of it, with $m < M$. We evaluate two settings for the $m \to M$ bottleneck: $64 \to 512$ and $512 \to 4096$.

Discrete bottleneck details. We use Finite Discrete Tokens (FDT) [2] to implement a discrete bottleneck. FDT encodes images and texts as discrete codes belonging to a shared learned codebook. During training, a convex combination of the discrete codes is produced using the Sparsemax operation [12], which the authors show performs better than Softmax. We follow the hyperparameters used by the authors: the codebook contains 16,384 codes of 512 dimensions each.

Results. We present our results in Tab. B.1. First, we see that a simple linear bottleneck performs worse than standard CLIP in a majority of settings, indicating that the mechanism behind the representational bottleneck is important. As expected, the discrete bottleneck widely outperforms or stays comparable to standard CLIP. However, note that global average pooling (GAP) continues to shine as a strong baseline, outperforming the discrete bottleneck in most settings. Finally, we verify that SPARO outperforms or remains comparable to the best baselines across all settings.

B.3 CLIP with ResNet encoders

We compare CLIP models with residual network (ResNet) [4] encoders, trained on CC15M with and without SPARO, in Tab. B.2. Our findings are consistent with those observed with ViT encoders, demonstrating that SPARO improves model performance across SugarCrepe, ImageNet, and robustness benchmarks without increasing model size.

B.4 Fine-grained SugarCrepe performance

The SugarCrepe [5] benchmark comprises of seven compositional manipulations of captions for a subset of MS COCO [10]. The benchmark evaluates the zero-shot accuracy of vision-language models for picking the ground-truth captions over the generated hard negatives. We provide the fine-grained results of our models for each of the seven SugarCrepe manipulations in Tab. B.3. Individual categories often demonstrate trade-offs amongst each other, but the averaged performance

Table B.3: Fine-grained SugarCrepe zero-shot accuracies for CLIP models trained on Conceptual Captions and LAION-400M.

Train	Model	Replace			Swap		Add		Avg
		Object	Attribute	Relation	Object	Attribute	Object	Attribute	
CC3M	CLIP^{16} ($\mathcal{C}$)	0.741	0.673	0.575	0.524	0.584	**0.615**	0.566	0.611
	$\mathcal{C}$+GAP	0.725	0.675	0.575	0.524	**0.598**	0.611	**0.601**	0.616
	$\mathcal{C}$+SPARO	**0.763**	**0.689**	**0.585**	**0.541**	0.592	0.611	0.592	**0.625**
CC12M	CLIP^{16} ($\mathcal{C}$)	0.860	0.737	**0.666**	0.593	0.619	0.743	0.659	0.697
	$\mathcal{C}$+GAP	0.873	0.747	0.647	0.573	0.619	0.741	**0.666**	0.695
	$\mathcal{C}$+SPARO	**0.881**	**0.784**	0.649	**0.642**	**0.674**	**0.768**	0.659	**0.723**
CC15M	CLIP^{16} ($\mathcal{C}$)	0.866	0.753	0.664	0.577	0.613	0.761	0.660	0.699
	$\mathcal{C}$+GAP	0.875	0.770	0.649	0.541	0.640	0.758	0.672	0.701
	$\mathcal{C}$+SPARO	**0.896**	**0.798**	**0.666**	**0.593**	**0.700**	**0.779**	**0.679**	**0.730**
L400M	CLIP^{32} ($\mathcal{C}$)	0.921	**0.826**	**0.678**	**0.605**	0.659	0.815	0.735	0.748
	$\mathcal{C}$+GAP	0.920	0.797	0.661	0.565	0.631	0.815	0.733	0.732
	$\mathcal{C}$+SPARO	**0.927**	0.822	**0.678**	0.585	**0.719**	**0.866**	**0.794**	**0.770**

reveals consistent trends of SPARO outperforming the baselines. Additionally, we show that the compositionality exposed in learned SPARO encodings across different categories can be enhanced by performing post-hoc concept selection, demonstrated in Sec. 5.1.

B.5 Winoground performance

We evaluate CLIP+SPARO against CLIP and CLIP+GAP on Winoground [19], and report the metrics suggested by the benchmark. Given image-caption pairs (I_0, C_0) and (I_1, C_1), the 'Text' score averages whether the model can select the correct captions C_0 and C_1 given the images I_0 and I_1 respectively. The 'Image' score tallies whether the model can select the correct images I_0 and I_1 for the captions C_0 and C_1 respectively. The 'Group' score combines these tests, and evaluates whether the model can select both the correct image given the caption, and the correct caption given the image, for combinations of image-text pairs. We report our results in Tab. B.4, showing superior performance of SPARO in all of the 'Image' and 'Group' evaluations. For 'Text', we find that SPARO performs as well as or better than GAP, but CLIP without GAP outperforms both in two out of our four training setups.

B.6 Absolute VTAB results

In Tab. B.5, we present the absolute numbers behind the relative improvements of SPARO illustrated in Fig. 2 for the VTAB benchmark [22].

Table B.4: Winoground zero-shot accuracies for CLIP models trained on Conceptual Captions and LAION-400M.

Train	Model	Text	Image	Group
CC3M	CLIP$^{16\ (\mathcal{C})}$	0.205	0.075	0.052
	$\mathcal{C}$+GAP	0.190	0.097	0.065
	$\mathcal{C}$+SPARO	**0.215**	**0.123**	**0.072**
CC12M	CLIP$^{16\ (\mathcal{C})}$	**0.248**	0.087	0.052
	$\mathcal{C}$+GAP	0.245	0.095	0.052
	$\mathcal{C}$+SPARO	0.245	**0.102**	**0.058**
CC15M	CLIP$^{16\ (\mathcal{C})}$	0.243	0.085	0.052
	$\mathcal{C}$+GAP	0.245	0.078	0.052
	$\mathcal{C}$+SPARO	**0.273**	**0.102**	**0.075**
L400M	CLIP$^{32\ (\mathcal{C})}$	**0.287**	0.110	0.083
	$\mathcal{C}$+GAP	0.265	0.132	0.087
	$\mathcal{C}$+SPARO	0.278	**0.140**	**0.097**

B.7 Additional visualizations

We present additional examples of learned SPARO concepts in Fig. B.2 following the filtering described in Sec. 5.4. Each row illustrates the attended positions of the single-head attention operation for the same SPARO slot across examples from MS COCO [10]. We reuse the same last input for the first two visualized slots to highlight cases where visually similar attention maps can correspond to distinct concepts. Looking at attended positions allows us to interpret *where* each slot takes information from, but not *how* that information is used.

C Reproducibility

We present the details of the training datasets used in our experiments in Tab. C.1. Note that the datasets we use for training our CLIP models are potentially smaller than those used by prior work, as images become unavailable for download over time for the datasets we consider. Tab. C.2 lists the open-source repositories we utilized and modified for training and for the evaluations presented in our work. Finally, note that Appendix A provides example PyTorch code for implementing the SPARO module.

C.1 CLIP

CLIP training. We use the open-source OpenCLIP [6] project for training our CLIP models. The important hyperparameters that we change from the OpenCLIP defaults for training our models are listed in Tab. C.3.

Table B.5: Absolute VTAB benchmark results corresponding to the relative differences illustrated in Fig. 2.

VTAB dataset	CC3M ($\mathcal{C}^{16}$)		CC12M ($\mathcal{C}^{16}$)		CC15M ($\mathcal{C}^{16}$)		L400M ($\mathcal{C}^{32}$)	
	GAP	SPARO	GAP	SPARO	GAP	SPARO	GAP	SPARO
cifar100	0.163	**0.168**	0.330	**0.390**	0.397	**0.449**	0.707	**0.742**
resisc45	0.169	**0.184**	0.326	**0.359**	0.357	**0.360**	0.527	**0.563**
eurosat	0.193	**0.226**	0.276	**0.335**	0.246	**0.292**	0.480	**0.513**
dtd	0.104	**0.121**	0.162	**0.234**	0.181	**0.195**	0.502	**0.506**
svhn	0.074	**0.097**	0.087	**0.098**	0.084	**0.125**	0.236	**0.458**
caltech101	0.539	**0.562**	0.731	**0.744**	**0.752**	0.748	0.828	**0.829**
smallnorb_elevation	0.114	**0.119**	0.100	**0.108**	**0.138**	0.116	0.102	**0.108**
smallnorb_azimuth	**0.055**	0.053	0.052	**0.053**	**0.058**	0.057	0.049	**0.053**
clevr_closest_obj	**0.224**	0.194	0.151	**0.197**	**0.109**	0.095	**0.204**	0.082
clevr_count	0.105	**0.132**	0.227	**0.232**	0.136	**0.151**	**0.189**	0.164
pcam	0.487	**0.495**	0.477	**0.535**	0.502	**0.572**	**0.549**	0.495
sun397	0.261	**0.263**	0.464	**0.489**	0.505	**0.519**	**0.671**	0.663
dmlab	0.148	**0.172**	**0.172**	0.135	0.134	**0.171**	**0.202**	0.141
kitti_closest_dist	0.118	**0.322**	**0.421**	0.392	0.249	**0.429**	**0.245**	0.125
diabetic_ret	0.066	**0.147**	**0.165**	0.118	0.166	**0.174**	**0.047**	0.023
dsprites_xpos	**0.033**	0.028	**0.030**	0.029	0.035	**0.044**	**0.032**	0.030
dsprites_orientation	**0.027**	0.020	**0.029**	0.026	**0.019**	0.015	0.017	**0.025**
pets	0.099	**0.104**	**0.587**	0.575	**0.595**	0.579	0.860	**0.869**
flowers	0.119	**0.131**	**0.298**	0.287	**0.324**	0.311	0.659	**0.673**

Linear probe training. We use the open-source CLIP benchmark [8] project for training and evaluating linear probes for our trained OpenCLIP checkpoints. We train our linear probes on ℓ_2-normalized CLIP encodings using the AdamW [11] optimizer with a learning rate of 0.1 for 10 epochs. For each setting, we first sweep for the optimal weight decay hyperparameter by training on a subset of the training split and evaluating on the remaining samples. Finally, we train on the full training set with the chosen weight decay value. The design of the hyperparameter sweep follows that of [15].

Training masks on SugarCrepe. Our training set is prepared by combining the splits of the SugarCrepe benchmark without any extra balancing. For each training sample, we obtain three encodings: the image encoding $\boldsymbol{y}^{\mathrm{i}}$, positive text encoding $\boldsymbol{y}^{\mathrm{p}}$, and the negative text encoding $\boldsymbol{y}^{\mathrm{n}}$. We produce the masked image encoding $\boldsymbol{y}^{\mathrm{i}} = \boldsymbol{m} \odot \boldsymbol{y}$, where $\boldsymbol{m}$ is generated as $\boldsymbol{m} = \sigma\left(\frac{1}{4}\max(100, \exp(\alpha))\,\boldsymbol{\theta}\right)$. Here, σ is the element-wise sigmoid function, $\alpha \in \mathbb{R}$ and $\boldsymbol{\theta} \in \mathbb{R}^{M'}$ are learned parameters, and $\max(100, \exp(\alpha))/4$ is the (inverse) temperature for the sigmoid that we found to help in faster convergence and stability of training. We have $M' = L$ for per-slot masking and $M' = M$ for per-dimension masking. We use the cosine similarities $\boldsymbol{y}^{\mathrm{i}} \cdot \boldsymbol{y}^{\mathrm{p}} / \|\boldsymbol{y}^{\mathrm{i}}\| \|\boldsymbol{y}^{\mathrm{p}}\|$ and $\boldsymbol{y}^{\mathrm{i}} \cdot \boldsymbol{y}^{\mathrm{n}} / \|\boldsymbol{y}^{\mathrm{i}}\| \|\boldsymbol{y}^{\mathrm{n}}\|$, scaled by $\max(100, \exp(\alpha))$ similar to CLIP [15], as the logits for the cross-entropy loss

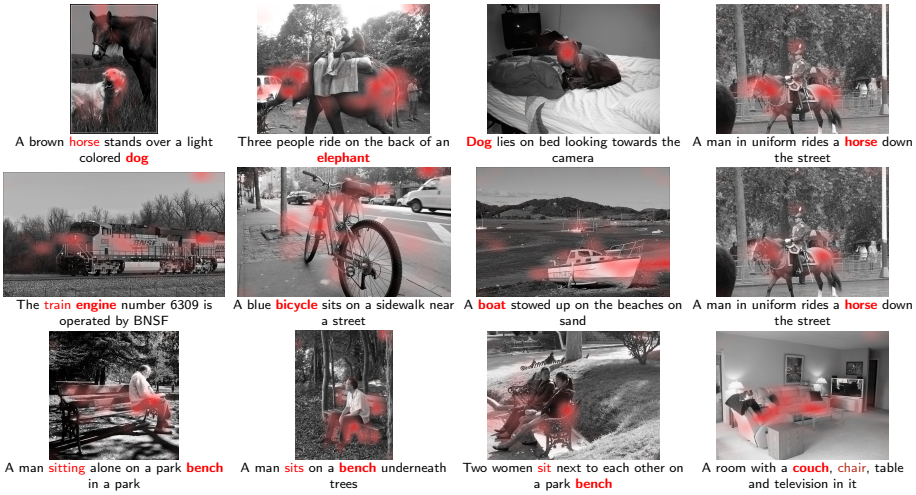


Fig. B.2: Additional visualizations of attended image and text positions for three SPARO slots (one per row). We surmise that the SPARO concepts from top to bottom represent animals, transportation, and seats. Notice that top-right and center-right examples have similar attention masks over 'horse,' but consider different aspects of the concept – one as an animal, another as a mode of transportation.

that we minimize. For each setting, we train for 100 epochs using SGD, with learning rate 0.02 and momentum 0.9. All settings converge before 100 epochs, and we pick the mask which achieves the best fit on standard SugarCrepe evaluation during training.

C.2 DINO

We use the official DINO repository [13] to train and evaluate our DINO models. For training, we follow the instructions for "Vanilla DINO training," using a ViT-Small backbone trained on a single node with 8 GPUs for 100 epochs. Similarly, we use the default hyperparameters for k-nearest neighbors and linear probe evaluations. The linear probe is trained on the layer normalized `CLS` token outputs from the last 4 ViT blocks. When training the linear probe on DINO+SPARO, we use the unnormalized SPARO encoding instead of the last block's normalized `CLS` output.

References

1. Changpinyo, S., Sharma, P., Ding, N., Soricut, R.: Conceptual 12M: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3558–3568 (2021)

Table C.1: Training datasets and the train split sizes used in our experiments. For CC3M, CC12M, and LAION-400M, our downloaded dataset sizes are smaller than the originally published sizes due to images becoming unavailable for download over time.

Training dataset	# Classes	Published size	Downloaded size	Download rate
CC3M [17]	N/A	3,318,333	2,795,293	84.24%
CC12M [1]		12,423,374	10,030,127	80.74%
CC15M (CC3M+CC12M)		N/A	12,825,420	N/A
LAION-400M [16]		413,871,335	319,707,672	77.25%
CIFAR-10 [7]	10	50,000		100%
CIFAR-100 [7]	100	50,000		100%
ImageNet [3]	1000	1,281,167		100%

Table C.2: Open source repositories used and their modifications for our experiments.

Task	Open-source repo	Summary of changes
CLIP Training CLIP ImageNet 0-shot	OpenCLIP [6]	Added SPARO and text AttPool
CLIP SugarCrepe 0-shot	SugarCrepe [20]	Support SPARO slot selection
CLIP linear probe CLIP 0-shot robustness CLIP 0-shot retrieval	CLIP benchmark [8]	
DINO Training DINO k-NN DINO linear probe	DINO [13]	Added SPARO

2. Chen, Y., Yuan, J., Tian, Y., Geng, S., Li, X., Zhou, D., Metaxas, D.N., Yang, H.: Revisiting multimodal representation in contrastive learning: from patch and token embeddings to finite discrete tokens. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15095–15104 (2023)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. IEEE (2009)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
5. Hsieh, C.Y., Zhang, J., Ma, Z., Kembhavi, A., Krishna, R.: Sugarcrepe: Fixing hackable benchmarks for vision-language compositionality. Advances in neural information processing systems (2023)
6. Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., Schmidt, L.: OpenCLIP. `https://github.com/mlfoundations/open_clip` (Jul 2021). `https://doi.org/10.5281/zenodo.5143773`
7. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Technical report (2009)

Table C.3: Hyperparameters for CLIP training using OpenCLIP [6].

Hyperparameter	Hyperparameter value			
	Dataset: CC3M	CC12M	CC15M	L400M
Epochs	32	25	25	32
Global batch size	4096			32768
Warmup steps	3600			2000
Weight decay	0.5			0.2
Precision	Automatic mixed precision with BFloat16			

8. LAION-AI: CLIP_benchmark open-source project. https://github.com/LAION-AI/CLIP_benchmark (2022)
9. Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., Teh, Y.W.: Set transformer: A framework for attention-based permutation-invariant neural networks. In: International conference on machine learning. pp. 3744–3753. PMLR (2019)
10. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: Fleet, D.J., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V. Lecture Notes in Computer Science, vol. 8693, pp. 740–755. Springer (2014). https://doi.org/10.1007/978-3-319-10602-1_48, https://doi.org/10.1007/978-3-319-10602-1%5F48
11. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
12. Martins, A., Astudillo, R.: From softmax to sparsemax: A sparse model of attention and multi-label classification. In: International conference on machine learning. pp. 1614–1623. PMLR (2016)
13. Meta Research: DINO open-source repository. https://github.com/facebookresearch/dino (2021)
14. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
15. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
16. Schuhmann, C., Vencu, R., Beaumont, R., Kaczmarczyk, R., Mullis, C., Katta, A., Coombes, T., Jitsev, J., Komatsuzaki, A.: Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. arXiv preprint arXiv:2111.02114 (2021)
17. Sharma, P., Ding, N., Goodman, S., Soricut, R.: Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers. pp. 2556–2565. Association for Computational Linguistics (2018)
18. Shazeer, N.: Fast transformer decoding: One write-head is all you need. arXiv preprint arXiv:1911.02150 (2019)

19. Thrush, T., Jiang, R., Bartolo, M., Singh, A., Williams, A., Kiela, D., Ross, C.: Winoground: Probing vision and language models for visio-linguistic compositionality. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5238–5248 (2022)
20. UW RAIVN Lab: SugarCrepe open-source repository. https://github.com/RAIVNLab/sugar-crepe (2023)
21. Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., Wu, Y.: Coca: Contrastive captioners are image-text foundation models. arXiv preprint arXiv:2205.01917 (2022)
22. Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A.S., Neumann, M., Dosovitskiy, A., et al.: A large-scale study of representation learning with the visual task adaptation benchmark. arXiv preprint arXiv:1910.04867 (2019)